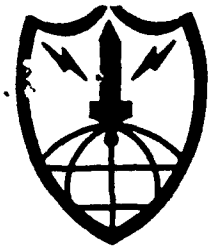

USAISEC

US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5300

U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES

The Software Support Qualitative Assessment Methodology

Volume II

The Review of Metrics for Developing an Information Systems Support Measurement Framework

March 1991

ASQB-GI-91-017

AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704--188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION / AVAILABILITY OF REPORT N/A
2b. DECLASSIFICATION / DOUWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ASQB-GI-91-017	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION The Center for Information Management Research	6b. OFFICE SYMBOL (if applicable)
6c. ADDRESS (City, State, and ZIP Code) SERC Georgia Institute of Technology Atlanta, GA 30332 Dept of MIS University of Arizona Tucson, AZ 85613	7a. NAME OF MONITORING ORGANIZATION N/A 7b. ADDRESS (City, State, and Zip Code) N/A
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AIRMICS	8b. OFFICE SYMBOL (if applicable) ASQB - GI
8c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg., Georgia Institute of Technology Atlanta, GA 30332-0800	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

10. SOURCE OF FUNDING NUMBERS			
PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
62783A	DY10	02-01-01	

11. TITLE (Include Security Classification) The Software Support Qualitative Assessment Methodology Volume II
The Review of Metrics for Developing an Information Systems Support Measurement Framework
(UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
W. Michael McCracken, Elizabeth Mynatt, Christopher Smith (GIT)
J.F. Nunamaker, Ai-Mei Chang, Titus Purdin, Richard Orwig, Amit Vyas (Univ of Arizona)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
final report	FROM _____ TO _____	1991, March, 22	23

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
life cycle metrics; systems support; support measures; qualitative assessment; supportability measures; assessment measures; readiness measures; information systems; software maintenance; support tools; tools management;

19. ABSTRACT (Continue on reverse if necessary and identity by block number)

The Software Supportability Qualitative Assessment Methodology is a five volume reference set that provides measures to aid in the support of information systems. The volumes are aimed at improving the support process by more accurately assessing the capabilities of support organizations, qualitatively measuring the supportability of fielded systems and evaluating the operational readiness of fielded systems. The five volumes are:

I. *Developing Quality Measures for Information Systems Support*
II. The Review of Metrics for Developing an Information Systems Support Measurement Framework
III. Implementing the Software Supportability Measure
IV. Implementing the Support Organization Assessment Measure
V. Implementing the Operational Readiness Measure

This volume provides a survey and evaluation of current metrics in terms of information sustems support. Specifically, three classes of metrics are reviewed: software product metrics, life cycle process metrics , and process management metrics.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED / UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c OFFICE SYMBOL
Howard C "Butch" Higley	(404) 894-3110	ASQB-GI

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

The research herein was performed for the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), the RDTE organization of the U.S. Army Information Systems Engineering Command (USAISEC). The sponsor for the project was the Office of the Director of Information Systems for Command, Control, Communications, and Computers (ODISC4). The principal investigator was from The Center for Information Management Research (CIMR), W. Michael McCracken of the Georgia Institute of Technology.

This research report is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited, and is not protected by copyright laws. Your comments on all aspects of the document are solicited.

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED

s/ _____
Glenn E. Racine
Chief
CISD

s/ _____
John R. Mitchell
Director
AIRMICS

The Software Support
Qualitative Assessment Methodology

Volume II

The Review of Metrics for Developing an Information Systems Support Measurement Framework

Prepared by

The Center for Information Management Research

for the

U.S. Army Institute for Research in Management Information, Communications, and Computer Science (AIRMICS)

Contract No. ECD-8904815

W. Michael McCracken, Elizabeth Mynatt, Christopher Smith

Software Engineering Research Center

Georgia Institute of Technology

December 1990

The **Software Supportability Qualitative Assessment Methodology** is a five volume reference set that provides measures to aid in the support of information systems. These manuals are aimed at improving the support process by more accurately assessing the capabilities of support organizations, quantitatively measuring the supportability of fielded systems and evaluating the operational readiness of fielded systems.

Volume I, *Developing Quality Measures for Information Systems Support*, describes the three measures along with the model of information system support that the measures are designed to satisfy. This is the main volume of the set and should be consulted before implementing the measures described in more detail in the other volumes.

Volume II, *The Review of Metrics for Developing an Information Systems Support Measurement Framework*, provides a survey and evaluation of current metrics in terms of information systems support. Specifically, three classes of metrics are reviewed: software product metrics, life cycle process metrics, and process management metrics.

Volume III, *Implementing the Software Supportability Measure*, provides instructions for collecting data for the measure, compiling the measure by evaluating the data, and interpreting the final result. The volume also contains guidelines for improving the supportabilty of an information system based on its evaluation. Specifically, the volume contains resource estimations for compiling and evaluating the measure, questionnaires for collecting the required data and step-by-step instructions for measuring the supportability of an information system.

Volume IV, *Implementing the Support Organization Assessment Measure*, provides instructions for collecting data for the assessment, conducting the assessment, and interpreting the final result. The volume also contains guidelines for improving the capabilities of a support organization based on its evaluation. Specifically, the volume contains resource estimations for conducting and evaluating the assessment, questionnaires for collecting the required data and step-by-step instructions for measuring the capabilities of a support organization.

Volume V, *Implementing the Operational Readiness Measure*, provides instructions for collecting data for the measure, compiling the measure by evaluating the data, and interpreting the final result. The volume also contains guidelines for improving the operational readiness of an information system based on its evaluation. Specifically, the volume contains resource estimations for compiling and evaluating the measure, questionnaires for collecting the required data and step-by-step instructions for measuring the operational readiness of an information system.

The Review of Metrics for Developing an Information Systems Support Measurement Framework

Christopher Smith
Elizabeth Mynatt

Center for Information Management Research
Georgia Institute of Technology
Atlanta, GA 30332-0280

1. Introduction

A major contributor to the life cycle cost of information systems is the cost of supporting these systems once they are fielded. The support of existing information systems has historically been perceived as a less glamorous task than the development of new information systems. However, it has been estimated that support costs now consume greater than 70 percent of total life cycle cost and that this proportion is still rising [24].

Unfortunately, while support cost has become an increasingly acute problem, there is not yet an effective means for determining support cost drivers and reducing support cost. Not only is the support process still poorly understood, there is no comprehensive set of quality measures that may be utilized to understand and control the process. Determining appropriate quality measures is no simple task, given the increasing size and complexity of information systems, the potentially large number of staff supporting these systems, the intricacy of information systems support planning, and the diversity of information systems users. The information systems support organization must have a method for evaluating their capability to adequately support their collection of information systems. In addition, measures of the supportability of information systems and the availability of these systems to the users is needed.

The objective of the Software Support Qualitative Assessment Methodology (contract no. ECD-8904815) is the development of a framework of measures for information systems support for utilization by the U. S. Army information systems support organizations. It is likely that a framework of measures for information systems support can be developed based on previously proposed life cycle measures. The key to developing this measurement framework is: (1) understanding the availability of current metrics; (2) constructing a valid model of the information systems support process; and (3) developing the measurement framework around the proposed model.

In this paper, we review some of the most important metrics that may be applied to the information systems support cycle, dividing the metrics into three broad classes. We then outline an information systems support model and describe top-level measures based on this model which are built from the reviewed metrics.

2. Classes of Measures for Information Systems Support

A *metric* is a measurable indication of some quantitative aspect of a system [10]. We use metrics to establish an indicator of merit for accurately evaluating software and its development and support. Metrics are also used to control and predict the quality of software and the process of developing and supporting this software. Metrics may be thought of as falling into one of three classes. One class of metrics quantifies attributes of the software product via static analysis of the software's source code and accompanying internal and external documentation Examples of attributes in this class are complexity and modularity. Another class of metrics quantifies aspects of the software development and support process. Still another class of metrics assesses the behavior or functionality of the software based upon its execution (e.g., a reliability metric).

These three classes of metrics can provide useful information to the various people involved in the information systems support process. The software technician is most interested in producing high-quality software and thus may find the static measures of product quality useful. The information systems support administrator interested in improving the support process may find the support process measures most useful. The user of the information systems is most interested in issues such as the usability, reliability, and availability of the information systems, thus suggesting the usefulness of behavioral metrics.

In the following three sections, a review of the three classes of metrics for information systems support is given. This review serves as foundation upon which we can subsequently build a framework of information systems support measures.

3. Software Product Metrics

Software product metrics measure characteristics exhibited by a static software product (source code, documentation). The metrics provide a method of measuring product quality. Although the product is a reflection of the process that created it, the metrics do not provide enough information to accurately analyze the process. Software product metrics are of particular importance to those most closely associated with the development and support of the product.

The characteristics of the software product which affect the ability to support the product are those determining the maintainability of the product. Maintainability is formally defined as follows:

> **Maintainability** is the ability of an item, under specified conditions of use, to be retained in, or restored to, within a given period of time, a specified state in which it can perform its required functions, when maintenance is performed under stated conditions and while using prescribed procedures and resources [9].

Modification of software is a non-trivial task. It involves such activities as program comprehension, diagnosis, repair or enhancement, and testing. Many software product design considerations affect the ease of software modification. These attributes are defined in the following table.

Complexity	A characteristic of the software interface which influences the resources another system will expend or commit while interfacing with the software [8].
Consistency	The extent to which uniform design techniques and notation are used [25].
Modularity	Characteristics which provide well-structured, highly cohesive, minimally coupled software [25].
Self-Descriptiveness	Characteristics which provide an explanation of the implementation of functions [25].
Testability	The extent to which software facilitates both the establishment of test criteria and the evaluation of the software with respect to those criteria [1].

Table 1: Software Design Factors Influencing Maintainability

Other aspects of the software product can affect its maintainability. Examples include the implementation language(s) and the size of the product. Another important factor in predicting the ability to perform corrective maintenance tasks on a software product is the age of the software, or, more directly, the extent to which the software has been previously modified. A recent study found that 83% of software faults were a result of modifications performed on an already fielded product [7], suggesting the crucial role post-release modification plays in the support of software.

To summarize, the set of factors that appear to most likely influence software maintainability is as follows:

Complexity	**Size**
Consistency	**Programming Language**
Modularity	**Age**
Self-Descriptiveness	**Modifications**
Testability	

Table 2 lists possible objective metrics that may be used to measure the attributes of software product maintainability. Not all attributes are listed in the table. Namely, metrics for consistency and self-descriptiveness, two attributes for which objective measurements are difficult to obtain, are not presented. In these two cases (and perhaps others where it may be difficult to gather objective measures), one would need to develop an additional set of subjective metrics to measure the attributes. For instance, we may measure the self-descriptiveness attribute by noting the existence and adequacy of certain types of documentation and the adherence of documentation to existing standards.

Attribute	*Metric*
Complexity	Cyclomatic Complexity [V(G)] [19] Effort Metric [14] Information Flow [17] Logical Stability [27] Nesting Level [8] Reachability [23] Logical Complexity [12] Span Between Data References [15]
Modularity	Cohesion Metric [11] Average Module Size
Testability	Reachability [23] Cohesion Metric [11]
Size	Lines of Code (LOC) Machine Lang. Instructions [22] Number of Tokens [14] Source Code Storage Space Object Code Storage Space Program Volume (V) [14] Module Count Average Module Size Procedure Count [5]
Language	Language Level [14]
Age	Program Age [5] Number of Program Releases [5]
Modifications	Total Number of Modifications Rate of Modifications

Table 2. Examples of Software Product Maintainability Metrics

From the examples presented in the Table 2, it is clear that there is a wide assortment of metrics from which to choose to measure software maintainability.

4. Life Cycle Process Metrics

Life cycle process metrics measure attributes of the process employed by an information systems organization to develop and support software. Since the software engineering process cannot be measured directly, we measure the elements directly contributing to the execution of the process. In other words, we measure the process *environment*. The setting of the process environment is the information systems support organization. Thus, life cycle process metrics usually measure some attribute of the development or support organization. In section 5 we discuss process metrics not *directly* measuring organizational attributes.

The ability to successfully execute the software engineering process depends on two key factors: the ability to successfully manage the process and the ability of resources to facilitate the successful execution of the process. The life cycle process metrics thus fall into one of the two broad categories of process management metrics and resource metrics.

4.1. Process Management Metrics

Process management metrics are the least well-defined of all software life cycle metrics. There are few, if any, concrete, objective management measures. As a result, most management measures are subjective, and their value or contribution to a high-level measure is difficult to accurately assess. A list of process management elements [21] are as follows:

- **Project Management**
  - **Organizational Structure**
  - **Life Cycle Planning**
  - **Design Methods**
  - **Implementation Methods**
  - **Testing Strategy**
  - **Project Interfaces**
- **Configuration Mangement**
  - **Identification**
  - **Control**
  - **Audit/Review**

A complete evaluation of the effectiveness a process management element involves the measurement of three characteristics of the element. These characteristics are **existence, usage, and adequacy.** The existence characteristic describes the presence or absence of an element of

process management believed to play a key contribution to the execution of the software process. The usage characteristic describes the degree of utilization of an existing process management element. The adequacy characteristic describes the effectiveness of the particular process management element that exists and is used.

Metrics for measuring process management characteristics are not as easily derivable as product metrics, since there is no tangible product to measure. The existence characteristic can be expressed as a binary measure, indicating whether a certain element exists or does not exist. The usage characteristic may be expressed in terms of a binary measure (used or not used), or it may be expressed in terms of the degree of an element's utilization, if such information is obtainable (via use of a rank scale or a utilization percentage). Adequacy is difficult to measure because it is completely subjective. One method of consistently measuring adequacy of an element is the employment of a rank scale (for instance, an integer scale of 0 to 5, where 0 = completely inadequate and 5 = completely adequate). An example of the use of such a scale is found in [20].

4.2. Resource Metrics

Resource metrics measure the capability of resources in helping to fulfill a support organization's ability to support its systems. The collection of resources is represented in the two-dimensional matrix shown in table 3. The top row of terms in this table denotes the various classes of resources, while the left hand column denotes characteristics we are interested in measuring. Thus, we have nine measures of interest: personnel availability, personnel utilization, and so forth.

	Personnel	**Software**	**Hardware**
Availability			
Utilization			
Capability			

Table 3: Resource Matrix

In constructing a complete set of resource measurements, if one wishes to utilize objective metrics to the maximal extent possible, then the best one may hope for is a mix of objective and subjective measures. Table 4 shows that, for some resource classes (in particular, anything involving personnel), many objective metrics have been proposed. On the other hand, metrics for many of the classes involving hardware or software are much harder to construct. Such metrics are dependent upon requirements and are likely to be at least in part subjective. In such cases, we once again would want to use the existence, usage and adequacy characteristics to obtain complete measures.

Characteristic	*Metric*
Hardware Utilization	CPU Utilization [22] I/O Channel Utilization [22] Memory Utilization [22] Test Resource Allocation [16]
Hardware Capability	Computer Storage Capacity
Personnel Availability	Number of Project Personnel [3] Number of Support Organization Personnel
Personnel Utilization	Staff-Hours [3] Percentage Staff Allocation for Task [22]
Personnel Capability	LOC/Production Period [22] Function Points [4] Education Level [3] Soft. Eng. Experience [3] Soft. Tech. Experience [3] Expert Availability [3] Organization Experience [3] Training [3] Project Experience [22] Productivity Quotient [26] Programmer Efficiency [26]
Software Utilization	Amount (time) Software Usage Frequency of Usage

Table 4: Examples of Resource Metrics

5. Behavioral Metrics

There is another class of metrics measuring attributes of the software product that may be analyzed only after the software has been dynamically executed. An example of such an attribute is software reliability. It is inappropriate to classify such metrics with the other life cycle process metrics because they are at best indirect measures of the software development and support process. Essentially these metrics measure the overall *effectiveness* of the software engineering process without actually measuring the process itself [13]. Table 5 lists the attributes which *behavioral metrics* measure. These attributes are adapted from [9,25], and [13].

Usability	The effort required to use software relative to the effort required to implement the software [25].
Reliability	The probability that software will not cause the failure of a system for a specified time under specified conditions [1].
Performance	A measure of the ability of a computer system to perform its functions [1].
Integrity	The extent to which software will perform without unauthorized access to code or data [25].
Availability	The probability that software will be able to perform its designated system function when required for use [1].

Some of the above attributes (reliability, in particular) have been studied in detail, and specific metrics have been proposed for these attributes. Others, such as usability, have not been as closely studied and are not as well specified. The difference between the two types of attributes appears to depend upon whether they are primarily objective or subjective measures. At one end of the spectrum, there are many objective metrics which may be applied to measure system reliability or availability. On the other hand, the usability of a system depends as much on human factors as on specified requirements (the usability metrics shown in table 6 are at best indirect measures of usability).

Attribute	*Metric*
Usability	Amount of User Training Frequency of Communication with Users Percentage of False Maintenance Requests
Reliability	Mean Time to Failure (MTTF) [2]. Failure Rate [2]. Fault Density [2]. Defect Density [2]. Estimated Number of Remaining Faults [2]. Testing Sufficiency [2]. Number of Maintenance Requests for Corrections Rate of Maintenance Requests for Corrections
Performance	Resource Consumption [13] Throughput [13] Response Time [13]
Integrity	System Access Control [6] System Access Identification [6]
Availability	Mean Time Between Failures [9] Mean Time to Repair [9] Total System Operational ("Up") Time [9] Total Time to Complete Maintenance Actions [9].

Table 6: Examples of Behavioral Process Metrics

6. Top-Level Measures of Information Systems Support

From the examples in the above sections it is apparent that there is a myriad of software engineering metrics from which one can choose. While a few metrics appear to have become more popular than the other metrics (for instance, certain metrics to measure the size and complexity of computer programs), there is no agreement upon which metrics truly provide the most accurate and important measures [28]. Additionally, the contribution of identifiable attributes to top-level measures such as software maintainability and supportability remains unclear. Empirical studies have thus far not resolved these issues. At any rate, the value of a particular metric is dependent upon the environment in which it is measured and the person(s) interpreting the resultant measure.

In our particular case, we would like to determine the appropriate metrics to incorporate into a framework for the assessment of information systems support. Before we can do so, we must develop a valid information systems support model around which we can center our measures.

6.1. Information Systems Support Model

The ability to support a software system involves a complex combination of factors, including product attributes, process attributes, and reliability attributes. The ability to develop and implement a support assessment methodology depends on the ability to successfully sort out the seemingly chaotic combination of support factors. We must also determine the perspectives for which we should base a high-level support measures.

Lientz and Swanson [18] identified three distinct populations in the information systems support environment:

- Information Systems Support Organization
- Information Systems
- Information Systems Users

A single high-level support measure may not accommodate all perspectives. The support organization is most likely concerned primarily with its ability to support its portfolio of software systems, suggesting the need for a organizational assessment measure. System users are more concerned with the availability or "operational readiness" of a software system, while the technical support staff maintaining a certain software product are particularly concerned with the supportability of that product. While each of the possible measures is influenced to a certain degree by various software product and process attributes, the impact of particular factors on these measures will vary.

A model of information systems support is shown in Figure 1. This model is an adaptation of the model presented in [24]. The model depicts the three distinct entities, or perspectives, within the information systems support environment. Depending upon one's perspective, the view of information systems support and its problems will differ. The lines connecting the entities in the model illustrate the relationships shared by the entities.

Two observations we may draw from the model to aid us in constructing support measures are:

- The view of information systems support varies depending upon one's perspective. Thus, certain top-level measures may be more valuable, and a certain set of low-level metrics may contribute significantly to the measures.
- Because there are relationships between the information systems, support organization, and collection of users, there will be at least a small degree of commonality among all top-level measures, even if such measures are interpreted differently.

In our model, we associate a top-level measure with each of the three perspectives. The three top-level measures are **supportability, support organization assessment,** and **operational readiness.** We define and discuss the motivations for the use of such terms in the following section.

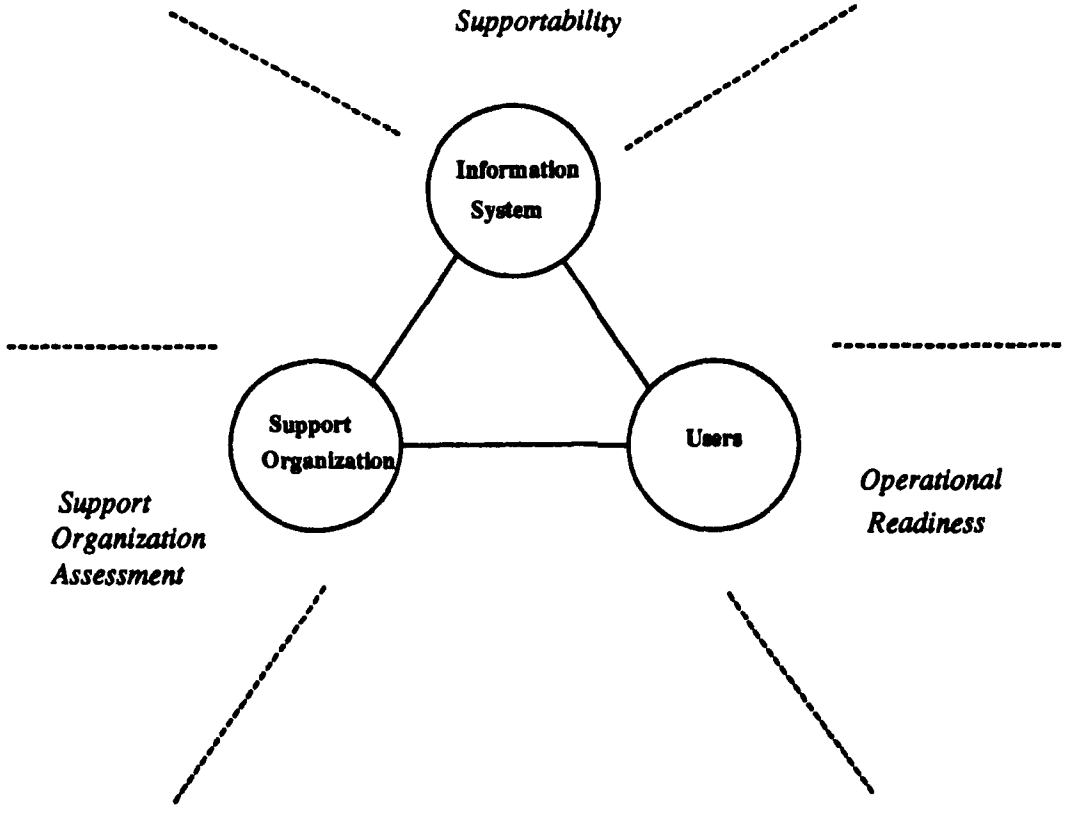

Figure 1: Support Model

6.2. Top-Level Support Measures

We have proposed three top-level information systems support measures to describe the information systems support environment while accommodating each of the support perspectives. From the information system's point of view, this measure is supportability. **Supportability** is the measure of the adequacy of products, resources, and procedures to facilitate:

- The intended operation of a software system or the restoration of the system to its intended operational state.
- The modification of the software system to meet new requirements.

For each information system supported, the supportability measure is intended to answer the questions, "Is the information system maintainable?", and, "Are the resources and procedures used to support the information system adequate?"

From the support organization's point of view, the measure we propose is the support organization assessment. A **support organization assessment** is an assessment, conducted either by the support organization or an outside agent, of the organization's ability to effectively support its information systems portfolio. The support organization assessment answers the question, "Can the support organization adequately keep its collection of information systems up and running?" Note that with this measure we treat information systems collectively rather than individually.

For the information system user's viewpoint, we propose operational readiness. **Operational readiness** is the ability of a software system to perform its intended function upon request, based on:

- The current operational state of the system.
- The reliability of the system.
- The supportability of the system.

Operational readiness addresses such questions as "Is the system up and running when I need it?", and, "When I use the system, can I expect correct results?" With operational readiness, we are again addressing information systems on an individual basis. In additionn, supportability (one of our other proposed measures) contributes to operational readiness, but user-oriented process measures also contribute to the top-level measure.

In previous sections we reviewed three classes of metrics for information systems support: product metrics, software engineering process metrics, and user-oriented process metrics. For each of our three top-level support measures, one class of metrics takes on primary importance. For the supportability measure, the maintainability product metrics are of primary importance, while the process metrics play a lesser role. The support organization is based heavily on support organization process metrics. The operational readiness measure is derived from the usage-oriented process metrics, although the other sets of metrics contribute to the measure as well.

6.3. Building the Top-Level Measures

The approach we take to develop the top-level support measures is a compromise between the incorporating extensive low-level metrics that would need to be collected to provide a complete albeit complex measure and the specification of only one or two metrics that supposedly would provide sufficient measurement of the entire support environment. On the one hand, the collection of a vast amount of metrics would be cumbersome and impractical (and probably unnecessary). On the other hand, it is unlikely that a single metric, such as a program complexity metric, wholly predicts the supportability of an information system.

We specify the composition of top-level measures in accompanying documents. The measures are developed using a combination of subjective and objective metrics, the objective metrics being partly drawn from the metrics outlined in the above sections. The rationale for selecting a particular metric to comprise a measure is based on the capability to gather the metric across a variety of information systems support environments and the predictive value of the metric in those environments.

7. Conclusion

In this paper, we reviewed existing metrics for measuring information systems support. We first divided the metrics into three classes. These classes of metrics serve as a basis from which we may develop a set of top-level support measures. Three top-level measures are proposed which accommodate the various perspectives in the information systems support environment modeled in Section 6. We may construct these measures by using a combination of objective and subjective metrics from each of the three metrics classes. In addition, the construction of the three top-level measures require differing interpretations of the role each class of metrics plays in the construction process. The combination of these top-level measures provide a complete view of the support environment and separately provide valuable information to the various support audiences. The development of these support measures is a necessary step in the understanding and control of the information systems support process. This ability to understand and control the process, in turn, leads to the greater ability to reduce information systems support costs and provide sustained support for information systems in the face of constantly changing user requirements.

8. Bibliography

1. *IEEE Standard Glossary of Software Engineering Terminology*, IEEE, New York (1983). Standard 729-1983

2. *IEEE Standard Dictionary of Measures to Produce Reliable Software*, IEEE, New York (1989). Standard 982.1-1988

3. *Standard for Software Productivity Metrics*, IEEE, New York (March 1990). Draft P1045/D2.1

4. C. A. Behrens, "Measuring the Productivity of Computer Systems Development Activities," *IEEE Transactions on Software Engineering* SE-9, pp. 648-652 (November 1983).

5. G. Benyon-Tinker, "Complexity Measures in an Evolving Large System," *Proceedings of the Workshop on Quantitative Software Models*, pp. 117-127, IEEE (October 1979).

6. T. P. Bowen, G. B. Wigle, and J. T. Tsal, "Specification of Software Quality Attributes: Software Quality Evaluation Handbook," RADC-TR-85-37, vols. I-III, Rome Air Development Center (February 1985).

7. J. S. Collofello and J. J. Buck, "Software Quality Assurance for Maintenance," *IEEE Software*, pp. 46-51 (September 1987).

8. S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*, Benjamin/Cummings, Menlo Park, CA (1986).

9. U. S. Department of Defense, "Test and Evaluation of System Reliability, Availability, and Maintainability: A Primer," DoD Directive 3235.1-H (March 1982).

10. T. DeMarco, *Controlling Software Projects*, Yourdon, New York (1982).

11. T. J. Emerson, "Program Testing, Path Coverage, and the Cohesion Metric," *Proceedings COMPSAC 84*, pp. 421-431, IEEE (1984).

12. T. Gilb, *Software Metrics*, Winthrop Publishers, Cambridge, MA (1977).

13. R. Guilfoyle, "Effectiveness Measures of the Software Process," *Proceedings of the Eighth Annual Conference on Ada Technology*, pp. 537-545 (March 1990).

14. M. H. Halstead, *Elements of Software Science*, Elsevier North-Holland, New York (1977).

15. W. Harrison, K. Magel, R. Kluczny, and A. DeKock, "Applying Software Complexity Metrics to Program Maintenance," *IEEE Computer* (September 1982).

16. W. Harrison, "Using Metrics to Allocate Testing Resources in a Resource Constrained Environment," TR 90-5, Portland State University (April 1990).

17. S. Henry and D. Kafura, "Software Structure Metrics Based on Information Flow," *IEEE*

Transactions on Software Engineering SE-7, pp. 510-518 (1981).

18. B. P. Lientz and E. B. Swanson, *Software Maintenance Management*, Addison-Wesley (1980).

19. T. J. McCabe, "A Complexity Measure," *IEEE Transactions on Software Engineering* SE-2, pp. 308-320 (December 1976).

20. D. E. Peercy and W. F. Huebner, "Risk Assessment Methodology for Software Supportability (RAMSS): Guidelines for Adapting Software Supportability Evaluations," BDM/ABQ-86-0090-TR, The BDM Corporation (April 1986).

21. D. E. Peercy, E. Tomlin, and G. Horlbeck, "Assessing Software Supportability Risk: A Minitutorial," *Proceedings - Conference on Software Maintenance*, pp. 72-80, IEEE (1987).

22. T. F. Saunders, "Software Reporting Metrics," MTR 9650, MITRE Software Center (April 1985).

23. N. F. Schneidewind and H.-M. Hoffman, "An Experiment in Software Error Data Collection and Analysis," *IEEE Transactions on Software Engineering* SE-5, pp. 276-285 (May 1979).

24. E. B. Swanson and C. M. Beath, *Maintaining Information Systems in Organizations*, New York, John Wiley and Sons (1989).

25. J. L. Warthman, "Software Quality Measurement Demonstration Project (I)," RADC-TR-87-247, Rome Air Development Center (December 1987).

26. D. Workman, "A Software Engineering Metric for Programmer Effectiveness," CS-TR-85-04, University of Central Florida (1985).

27. S. S. Yau and S.-C. Chang, "Estimating Logical Stability in Software Maintenance," *Proceedings COMPSAC 84*, pp. 109-119, IEEE (1984).

28. J. C. Zolnowski and D. B. Simmons, "Taking the Measure of Program Complexity," *Proceedings of the National Computer Conference*, pp. 329-336 (1981).

References

[CDS86] S. D. Conte, H. E. Dunsmore, and V. Y. Shen. *Software Engineering Metrics and Models*. Benjamin/Cummings, Menlo Park, CA, 1986.

[DeM82] T. DeMarco. *Controlling Software Projects*. Yourdon, New York, 1982.

[Dep82] Department of Defense. *Test and Evaluation of System Reliability, Availability, and Maintainability: a Primer*, March 1982. DoD Directive 3235.1-H.

[IEE83] IEEE. *IEEE Standard Glossary of Software Engineering Terminology*, 1983. Standard 729-1983.

[PTH87] D. E. Peercy, E. Tomlin, and G. Horlbeck. Assessing software supportability risk: A minitutorial. In *Conference on Software Maintenance*, pages 72–80. IEEE, 1987.

[Sch90] Stephen R. Schach. *Software Engineering*. Aksen Associates Incorporated Publishers, Boston, MA, 1990.

[Som89] Ian Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, New York, 1989.

[War87] J. L. Warthman. Software quality measurement demonstration project (i). Technical Report RADC-TR-87-247, Rome Air Development Center, December 1987.